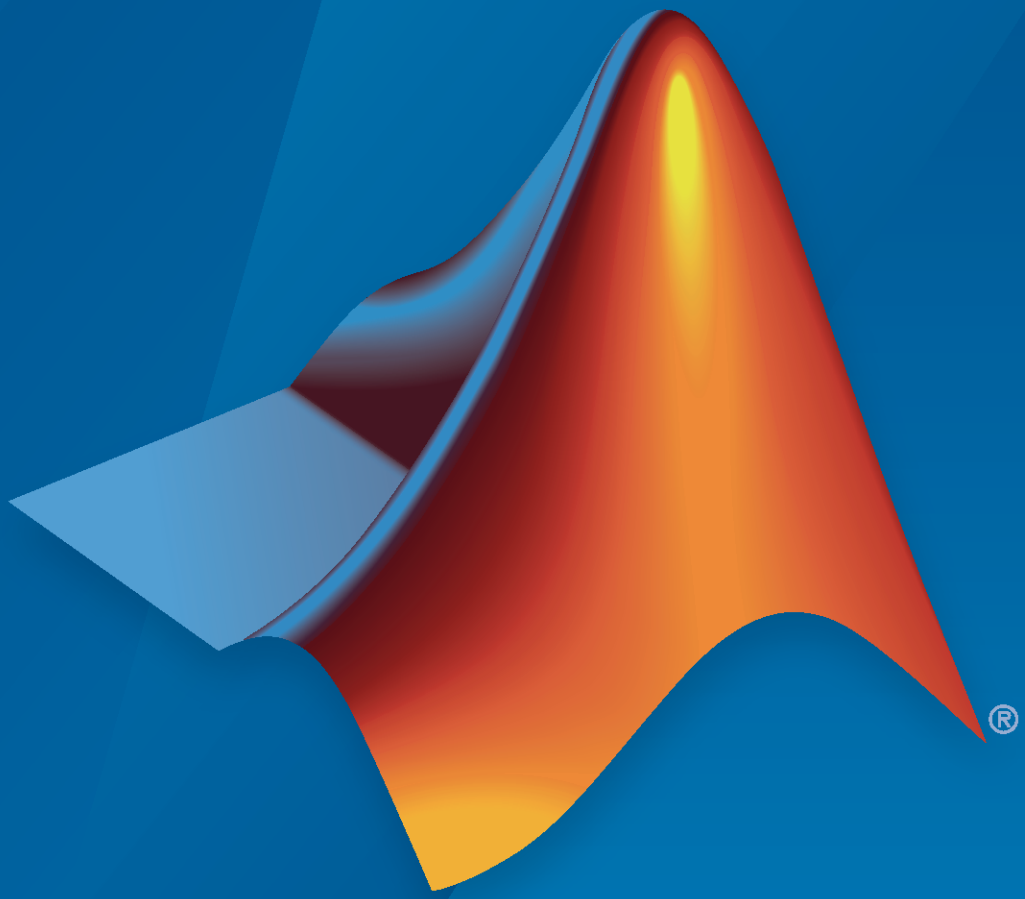
Text Analytics Toolbox™

Getting Started Guide



MATLAB®

R2023a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Text Analytics Toolbox™ Getting Started Guide

Revision History

September 2017	Online Only	New for Version 1.0
March 2018	Online Only	Revised for Version 1.1 (Release 2018a)
September 2018	Online Only	Revised for Version 1.2 (Release 2018b)
March 2019	Online Only	Revised for Version 1.3 (Release 2019a)
September 2019	Online Only	Revised for Version 1.4 (Release 2019b)
March 2020	Online Only	Revised for Version 1.5 (Release 2020a)
September 2020	Online Only	Revised for Version 1.6 (Release 2020b)
March 2021	Online Only	Revised for Version 1.7 (Release 2021a)
September 2021	Online Only	Revised for Version 1.8 (Release 2021b)
March 2022	Online Only	Revised for Version 1.8.1 (Release 2022a)
September 2022	Online Only	Revised for Version 1.9 (Release 2022b)
March 2023	Online Only	Revised for Version 1.10 (Release 2023a)

Contents

1 Getting Started

Getting Started

Text Analytics Toolbox Product Description

Analyze and model text data

Text Analytics Toolbox provides algorithms and visualizations for preprocessing, analyzing, and modeling text data. Models created with the toolbox can be used in applications such as sentiment analysis, predictive maintenance, and topic modeling.

Text Analytics Toolbox includes tools for processing raw text from sources such as equipment logs, news feeds, surveys, operator reports, and social media. You can extract text from popular file formats, preprocess raw text, extract individual words, convert text into numerical representations, and build statistical models.

Using machine learning techniques such as LSA, LDA, and word embeddings, you can find clusters and create features from high-dimensional text datasets. Features created with Text Analytics Toolbox can be combined with features from other data sources to build machine learning models that take advantage of textual, numeric, and other types of data.

Key Features

- Text preprocessing and normalization
- Machine learning algorithms, including latent Dirichlet allocation (LDA) and latent semantic analysis (LSA)
- Word-embedding training, and pretrained model import from word2vec, FastText, and GloVe
- Word cloud and text scatter plots
- Document import from PDF and Microsoft® Word files
- TF-IDF and word frequency statistics

Try Text Analytics in 10 Lines of Code

This example shows how to use text analytics to classify text data using only 10 lines of MATLAB® code. Try the example to see how simple it is to get started with text analytics in MATLAB.

You can create a simple classification model which uses word frequency counts as predictors. This example trains a classification model to predict the event type of factory reports using text descriptions.

Create Model

The main steps of creating a model are:

1. Import – import the text data into MATLAB.
2. Preprocess – preprocess the text for word analysis.
3. Convert – convert the text to numeric data.
4. Train – train a classification model.

Import the example text data and labels, tokenize the text, convert it to numeric data using a bag-of-words model, and train a supervised SVM classifier.

```
data = readtable('factoryReports.csv','TextType','String'); % Read data
labels = categorical(data.Category);                        % Read labels

documents = tokenizedDocument(data.Description);            % Preprocess text

bag = bagOfWords(documents);                                % Count words
XTrain = bag.Counts;                                        % Convert to numeric data

mdl = fitcecoc(XTrain,labels,'Learners','linear');          % Train classifier
```

Predict Using New Data

The steps for prediction are similar to those for training. To predict using new data, preprocess the text data and convert it to numeric using the same steps used for training. Then, predict the label using the trained model.

Predict the label for the text `"Coolant is pooling underneath sorter."`.

```
str = "Coolant is pooling underneath sorter.";              % Import text
documentsNew = tokenizedDocument(str);                      % Preprocess text
XTest = encode(bag,documentsNew);                           % Convert to numeric
label = predict(mdl,XTest)                                  % Predict label
```

```
label = categorical
     Leak
```

For an example showing a more detailed workflow, see “Create Simple Text Model for Classification”.

For next steps in text analytics, you can try improving the model accuracy by preprocessing the data and visualize the text data using word clouds. For examples, see “Prepare Text Data for Analysis” and “Visualize Text Data Using Word Clouds”.

See Also

`tokenizedDocument` | `bagOfWords` | `encode`

Related Examples

- “Analyze Text Data Using Topic Models”
- “Analyze Text Data Using Multiword Phrases”
- “Analyze Text Data Containing Emojis”
- “Train a Sentiment Classifier”
- “Classify Text Data Using Deep Learning”
- “Generate Text Using Deep Learning” (Deep Learning Toolbox)

Import Text Data into MATLAB

Use the following functions to import text data into MATLAB®. Usually, the easiest way to import text data into MATLAB is to use the `extractFileText` function. For example, to import text from a text file, use:

```
str = extractFileText('sonnets.txt');
```

This table outlines which function to use for different file types. For more information, see “Extract Text Data from Files”.

Source	Function	Example
Text, Microsoft Word, PDF, and HTML files	`extractFileText`	`str = extractFileText('sonnets.txt');`
CSV and Microsoft Excel® files	`readtable`	`tbl = readtable('weatherReports.csv','TextType','string');`
HTML code	`extractHTMLText`	`str = extractHTMLText('<html><body><h1>THE SONNETS</h1><p>by William Shakespeare</p></body></html>');`
PDF forms	`readPDFFormData`	`data = readPDFFormData('weatherReportForm1.pdf');`

See Also

`extractFileText` | `readtable` | `extractHTMLText` | `readPDFFormData`

More About

- “Try Text Analytics in 10 Lines of Code” on page 1-3
- “Create Simple Preprocessing Function” on page 1-6
- “Get Started with Topic Modeling” on page 1-12
- “Visualize Text Data Using Word Clouds”

Create Simple Preprocessing Function

This example shows how to create a function which cleans and preprocesses text data for analysis using the **Preprocess Text Data** Live Editor task.

Text data can be large and can contain lots of noise which negatively affects statistical analysis. For example, text data can contain the following:

- Variations in case, for example "new" and "New"
- Variations in word forms, for example "walk" and "walking"
- Words which add noise, for example "stop words" such as "the" and "of"
- Punctuation and special characters
- HTML and XML tags

These word clouds illustrate word frequency analysis applied to some raw text data from weather reports, and a preprocessed version of the same text data.

Most workflows require a preprocessing function to easily prepare different collections of text data in the same way. For example, when you train a model, you can use the same function to preprocess the training data and new data using the same steps.

You can interactively preprocess text data using the **Preprocess Text Data** Live Editor task and visualize the results. This example uses the **Preprocess Text Data** Live Editor task to generate code

that preprocesses text data and creates a function that you can reuse. For more information on Live Editor tasks, see “Add Interactive Tasks to a Live Script”.

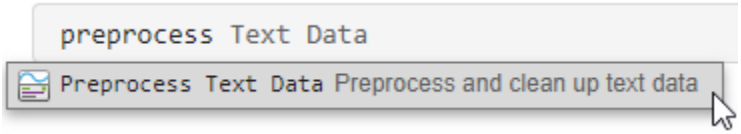
First, load the factory reports data. The data contains textual descriptions of factory failure events.

```
tbl = readtable("factoryReports.csv")
```

`tbl = 480×5 table`

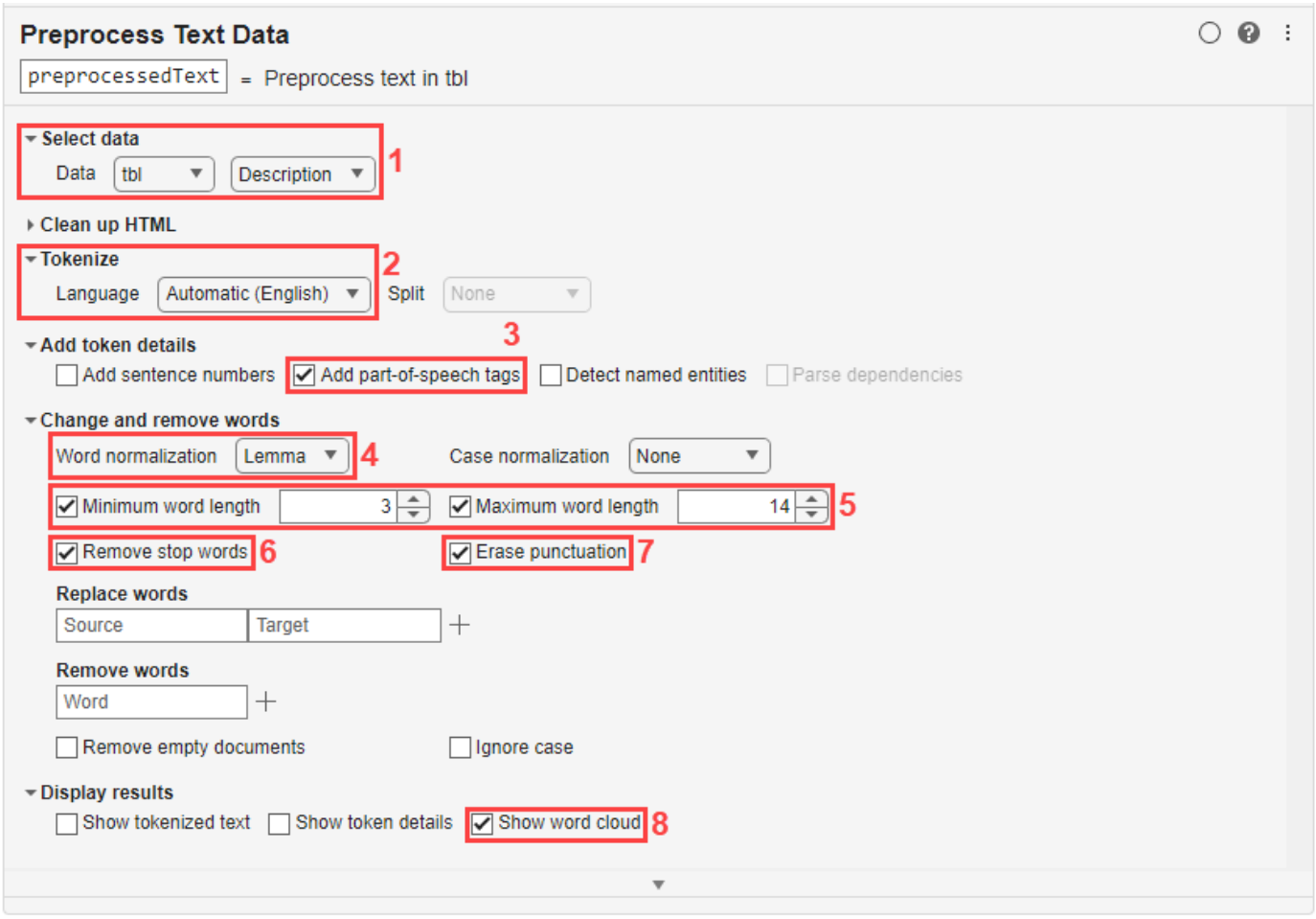
	Description	Category	Urgency
1	'Items are occasionally getting stuck in the scanner spools.'	'Mechanical Failure'	'Medium'
2	'Loud rattling and banging sounds are coming from assembler pistons.'	'Mechanical Failure'	'Medium'
3	'There are cuts to the power when starting the plant.'	'Electronic Failure'	'High'
4	'Fried capacitors in the assembler.'	'Electronic Failure'	'High'
5	'Mixer tripped the fuses.'	'Electronic Failure'	'Low'
6	'Burst pipe in the constructing agent is spraying coolant.'	'Leak'	'High'
7	'A fuse is blown in the mixer.'	'Electronic Failure'	'Low'
8	'Things continue to tumble off of the belt.'	'Mechanical Failure'	'Low'
9	'Falling items from the convevor belt.'	'Mechanical Failure'	'Low'

Open the **Preprocess Text Data** Live Editor task. To open the task, begin typing the keyword `preprocess` and select **Preprocess Text Data** from the suggested command completions. Alternatively, on the **Live Editor** tab, select **Task** > **Preprocess Text Data**.



Preprocess the text using these options:

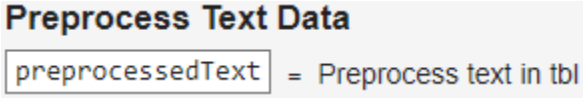
1. Select `tbl` as the input data and select the table variable `Description`.
2. Tokenize the text using automatic language detection.
3. To improve lemmatization, add part-of-speech tags to the token details.
4. Normalize the words using lemmatization.
5. Remove words with fewer than 3 characters or more than 14 characters.
6. Remove stop words.
7. Erase punctuation.
8. Display the preprocessed text in a word cloud.


Preprocess Text Data
preprocessedText = Preprocess text in tbl
Select data
Data tbl Description
1
Clean up HTML
Tokenize
Language Automatic (English)
Split None
2
3
Add token details
Add sentence numbers
Add part-of-speech tags
Detect named entities
Parse dependencies
Change and remove words
Word normalization Lemma
4
Case normalization None
Minimum word length 3
Maximum word length 14
5
Remove stop words
6
Erase punctuation
7
Replace words
Source
Target
Remove words
Word
Remove empty documents
Ignore case
Display results
Show tokenized text
Show token details
Show word cloud
8

The **Preprocess Text Data** Live Editor task generates code in your live script. The generated code reflects the options that you select and includes code to generate the display. To see the generated code, click the ▾ at the bottom of the task parameter area. The task expands to display the generated code.

```
%% Preprocess Text
preprocessedText = tbl.Description;

% Tokenize
preprocessedText = tokenizedDocument(preprocessedText);

% Add token details
preprocessedText = addPartOfSpeechDetails(preprocessedText);

% Change and remove words
preprocessedText = normalizeWords(preprocessedText,Style="lemma");
preprocessedText = removeShortWords(preprocessedText,2);
preprocessedText = removeLongWords(preprocessedText,15);
preprocessedText = removeStopWords(preprocessedText,IgnoreCase=false);
preprocessedText = erasePunctuation(preprocessedText);

% Display results
wordcloud(preprocessedText);
```

By default, the generated code uses `preprocessedText` as the name of the output variable returned to the MATLAB workspace. To specify a different output variable name, enter a new name in the summary line at the top of the task.



To reuse the same steps in your code, create a function that takes as input the text data and outputs the preprocessed text data. You can include the function at the end of a script or as a separate file. The `preprocessTextData` function listed at the end of the example, uses the code generated by the **Preprocess Text Data** Live Editor task.

To use the function, specify the table as input to the `preprocessTextData` function.

```
documents = preprocessTextData(tbl);
```

Preprocessing Function

The `preprocessTextData` function uses the code generated by the **Preprocess Text Data** Live Editor task. The function takes as input the table `tbl` and returns the preprocessed text `preprocessedText`. The function performs the these steps:

1. Extract the text data from the `Description` variable of the input table.
2. Tokenize the text using `tokenizedDocument`.
3. Add part-of-speech details using `addPartOfSpeechDetails`.
4. Lemmatize the words using `normalizeWords`.
5. Remove words with 2 or fewer characters using `removeShortWords`.
6. Remove words with 15 or more characters using `removeLongWords`.
7. Remove stop words (such as "and", "of", and "the") using `removeStopWords`.
8. Erase punctuation using `erasePunctuation`.

```
function preprocessedText = preprocessTextData(tbl)

%% Preprocess Text
preprocessedText = tbl.Description;

% Tokenize
preprocessedText = tokenizedDocument(preprocessedText);

% Add token details
preprocessedText = addPartOfSpeechDetails(preprocessedText);

% Change and remove words
preprocessedText = normalizeWords(preprocessedText,Style="lemma");
preprocessedText = removeShortWords(preprocessedText,2);
preprocessedText = removeLongWords(preprocessedText,15);
preprocessedText = removeStopWords(preprocessedText,IgnoreCase=false);
preprocessedText = erasePunctuation(preprocessedText);

end
```

For an example showing a more detailed workflow, see “Preprocess Text Data in Live Editor”. For next steps in text analytics, you can try creating a classification model or analyze the data using topic

models. For examples, see “Create Simple Text Model for Classification” and “Analyze Text Data Using Topic Models”.

See Also

Preprocess Text Data | `tokenizedDocument` | `erasePunctuation` | `removeStopWords` | `removeShortWords` | `removeLongWords` | `normalizeWords` | `addPartOfSpeechDetails`

More About

- “Preprocess Text Data in Live Editor”
- “Try Text Analytics in 10 Lines of Code” on page 1-3
- “Import Text Data into MATLAB” on page 1-5
- “Get Started with Topic Modeling” on page 1-12
- “Visualize Text Data Using Word Clouds”

Get Started with Topic Modeling

This example shows how to fit a topic model to text data and visualize the topics.

A Latent Dirichlet Allocation (LDA) model is a topic model which discovers underlying topics in a collection of documents. Topics, characterized by distributions of words, correspond to groups of commonly co-occurring words. LDA is an unsupervised topic model which means that it does not require labeled data.

Load and Extract Text Data

Load the example data. The file `factoryReports.csv` contains factory reports, including a text description and categorical labels for each event.

Import the data using the `readtable` function and extract the text data from the `Description` column.

```
filename = "factoryReports.csv";
data = readtable(filename,'TextType','string');
textData = data.Description;
```

Prepare Text Data for Analysis

Tokenize and preprocess the text data and create a bag-of-words model.

Tokenize the text.

```
documents = tokenizedDocument(textData);
```

To improve the model fit, remove the punctuation and the stop words (words like "and", "of", and "the") from the documents.

```
documents = removeStopWords(documents);
documents = erasePunctuation(documents);
```

Create a bag-of-words model.

```
bag = bagOfWords(documents);
```

Fit LDA Model

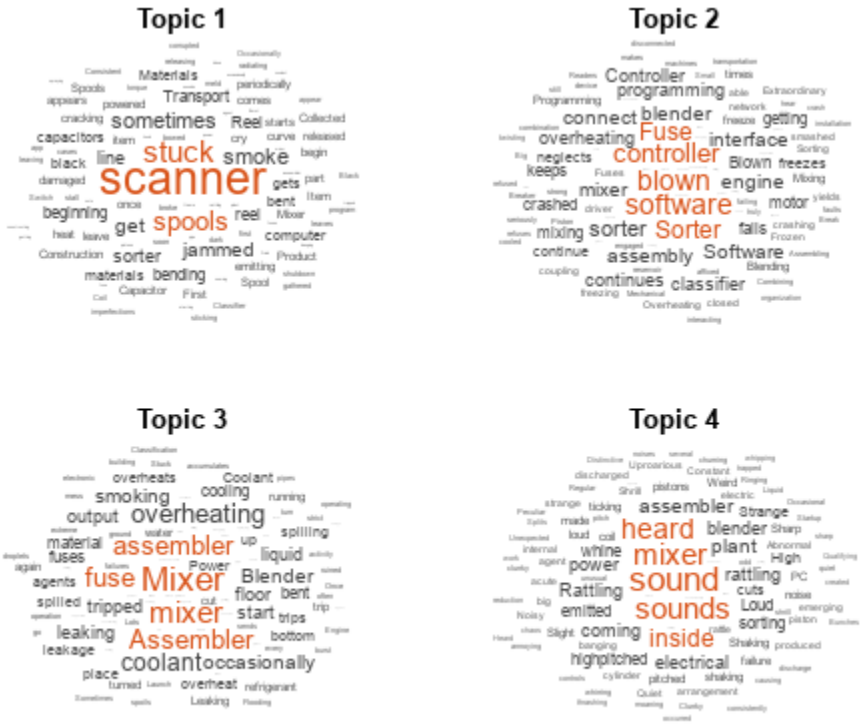
Fit an LDA model with seven topics using the `fitlda` function. To suppress the verbose output, set the `'Verbose'` option to `0`.

```
numTopics = 7;
mdl = fitlda(bag,numTopics,'Verbose',0);
```

Visualize Topics

Visualize the first four topics using word clouds.

```
figure
for topicIdx = 1:4
    subplot(2,2,topicIdx)
    wordcloud(mdl,topicIdx);
    title("Topic " + topicIdx)
end
```



For next steps in text analytics, you can try improving the model fit by using different preprocessing steps and visualizing the topic mixtures. For an example, see “Analyze Text Data Using Topic Models”.

See Also

`removeStopWords` | `tokenizedDocument` | `erasePunctuation` | `bagOfWords` | `fitlda` | `wordcloud`

More About

- “Try Text Analytics in 10 Lines of Code” on page 1-3
- “Import Text Data into MATLAB” on page 1-5
- “Create Simple Preprocessing Function” on page 1-6
- “Visualize Text Data Using Word Clouds”